The University of Birmingham
School of Computer Science
MSc in Advanced Computer Science

Second semester mini-project

Analysing User Interaction On Mobile Devices

Karthikeya Udupa Kuppar Manjunath

Supervisor: Mirco Musolesi

April 2014

Abstract

Today mobile phones are no longer a lifestyle choice but have become more of a necessity with a constant presence in our everyday life; these devices not only help in solving our day to day problems but also help us in the betterment of our lives. We constantly interact with these devices for assistance and information. But can the device act before the user asks for help? Can it be self-aware and understand its user and his behaviour? These are the few questions we are trying to answer through this research. We try to analyse the components involved in designing of such an *Anticipatory system* and why context forms an integral part of such systems. We then present *WebSense*, an android application developed for the purpose of monitoring and recording user interaction on mobile phones to enable future research to understand user behaviour on smartphones and the various factors that influence it. In order to gather contextual information and to provide a foundation for future work in this domain. We present easy to use and extensible Android Context Sensing Framework. *WebSense* gathers and synchronises all the information that is available onto the server providing a platform for a large scale research with a diverse user-base and a very rich contextual database. We try to understand the challenges faced in designing an anticipatory systems of this nature and also evaluate the present system. We look into the work that has already been done in this domain and a few possible applications of the data collected by the application and analyse them further in details.

Keywords

Context, Context-Awareness, Anticipatory system, Ambient intelligence, Context Sensing Framework, Android, Smartphone usage, Activity monitoring

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

In the recent years there has been a phenomenal growth in mobile and smartphone technology. The world has been revolutionised by technologies beyond what one could have even imagined a century ago. The digital revolution has effected our life's every possible aspect. Mobility being one of the key components of this revolution and has become a very important aspect in the lives of its consumers(Ross 2011) and is being adopted by even more people worldwide at a drastic rate. Smartphones and other portable devices have become an extension of our persona and have been personalised by its users. It is a platform which receives a glimpse into the lives of its users, our likes and dislikes, social choices we make, various minute preferences and decisions that we take, places we travel and even the food we prefer to eat can all be understood by how one uses his device. These devices have provided a new avenue to move forward with what Mark Wesiser envisioned two decades ago in this revolutionary article (Weiser 1991). In addition to this it also provides us a means to achieve more in terms of understanding user behaviour. Through its vast and rapidly expanding network of wireless sensors which is always in hands of its users, provides a limitless source of contextual information (Campbell, Eisenman, Lane, Miluzzo, Peterson, Lu, Zheng, Musolesi, Fodor, and Ahn 2008) which then would help in performing studies which would help the user can devote his time into real world rather then being tangled in technological complexities.

The goal of this project is to understand and develop a foundation for a system which would help us in monitoring, processing and learning about how the end users interact with their mobile devices. It also looks into the various factors that constitute a vital role in designing an effective anticipatory system which can provide information beforehand hence improving the overall user experience and making the lives of its user a bit less complex. A key aspect for such an application would be for it to be ubiquitous and to provide a solution which would somehow help improve their lives and not be a hindrance in their day to day task. It would be anticipating their actions and adapt itself to their needs, Tennehouse(Tennenhouse 2000) introduces this concept as proactive computing wherein the system would become so familiar with the user that it would be able to perform action on user's behest.

For us to accomplish this we would require various components, foundations of some of which we are building as a part of this research. One of such component is a way in which we can monitor what activities people perform on their devices. We would also require information about what are the various contextual conditions which may have an impact on whatever action user performs. All this would have be done at a very large scale in order to get a more diverse demographics of application usage and contextual information so that the system could cater more users once its complete. This report introduces the Android context sensing framework and the *WebSense* application which will try to accomplish some of the above mentioned tasks. The development process for such a system faces some serious challenges and concerns which we will discuss as well and try to find solutions for them.

1.2 Context & Context-awareness

To fully understand the purpose and the scope of this research it is very essential for one to understand what Context and context-awareness means. Context provides us with insight about the factors that lead the user in performing a certain action on his mobile device. Oxford dictionary defines context as "the circumstances that form the setting for an event, statement, or idea, and in terms of which it can be fully understood". Over the years there have been many efforts to define context in terms of computing (Schilit, Adams, and Want 1994; Brown 1996; Prekop and Burnett 2003), each with their own parameters and adding further specifications to consider to the previous definitions. With the rapid pace of development of technology, we can say that each definition eventually starts to seem incomplete after a certain period of time. In the *WebSense* application context can refer to the any information that can define the condition of the user while an interaction occurs between him and the device. This also covers the relevant state of the device along with the context (connectivity, current power level, nearby connected devices among others). (Dey 2001) defines Context as "any information that can be used to characterise the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves." which very clearly defines what we are trying to accomplish through the application in which case it constitutes of physical, computing and time contexts(Schilit, Adams, and Want 1994).

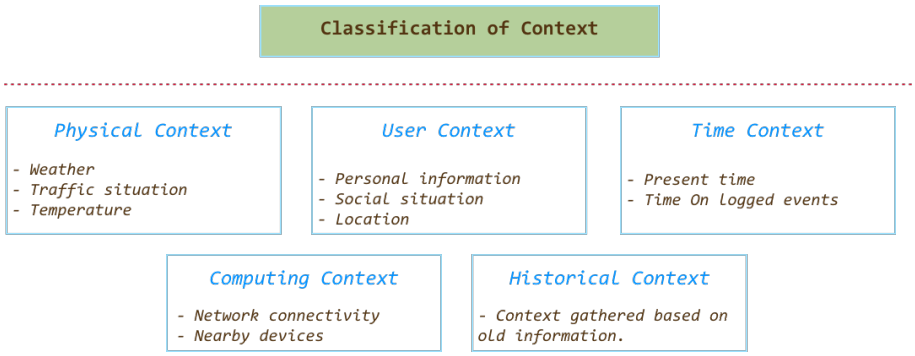


Figure 1.1: Classification of context information as per (Schilit, Adams, and Want 1994)

Context Awareness on the other hand is the most basic building block for an anticipatory system. It specifies the ability of the system to know about the state in which it is presently in and eventually adapt itself to the conditions based on the knowledge it has gained over the period time about the user's choices. Context-awareness acts as an extension of human senses, fulfilling its limitation, helping in continuously understanding the user's context and improvising itself in its ability to help the user perform his tasks more effectively. Similarly as with all other computing problems, context awareness has moved far ahead from the initial description of context(Schilit and Theimer 1994) with the emergence of readily available powerful sensing devices in the hands of masses in the form of mobile smartphones hence allowing us to move ahead far more than one had imagined earlier.

Chapter 2

Ambient Intelligence & Anticipatory System

The field of ambient intelligence and anticipatory systems are closely interrelated with each other. Ambient intelligence acts as a foundation for systems which want to perform anticipatory tasks. Both can actually be considered as components of a system which Weiser might have envisioned when he said "The most profound revolutions are not the ones trumpeted by pundits, but those that sneak in when we are not looking"(Weiser 1991).

The term *Ambient Intelligence* was first introduced in the AmI Challenge in 2001 (Ducatel, Bogdanowicz, Scapolo, Leijten, and Burgelman 2001), wherein it identified Ambient Intelligence at a conceptual level, highlighting the various components that would be required to make it a reality. (Sadri 2011) defines it as an environment wherein the the environment itself would adapt to the needs of the user, there would be a lack of any input device to take user's feedback instead there would be sensors which would blend in into the environment's day to day items, connected with each other and constantly trying to improvise and creating a positive impact in the life of the user.

Anticipatory system is described as "a system containing a predictive model of itself and/or of its environment, which allows it to state at an instant in accord with the model's predictions pertaining to a later instant" (Rosen 2012). Hence, the system has to be intelligent enough to make predictions about the future and also to adapt itself, this makes it very similar to ambient intelligence, the difference being that the ambience of the technology which allows it to render invisible to the user may or may not be there in the anticipatory system. The current framework and application mentioned further in this report form the ground work for a system which falls into the category of both ambient intelligence and anticipatory system.

2.1 Components Of an anticipatory system

An anticipatory system constitutes of multiple components which work in a synchronous manner to provide the desired end result to its users. As we have seen that the context for any action tells us a lot about the user, the action being performed and the reasons behind it, hence forming an essential part of any system that deals with understanding and improving user's day to day

tasks and habits. Majority of the information required to do such a task is extracted from the sensors that are available on the devices, however the raw data has very limited use and usually bound to an individual. We require a larger dataset consisting of multiple individuals to attain the desirable results. After collection of raw data features are extracted from it and meaningful information is obtained helping us derive inferences. These inferences can then be used to predict future events and action, gradually improving itself and its ability to predict over a period of time in the form of reinforced learning. (Pejovic and Musolesi 2013) categorises the entire system into four staged process:

- Sensing of the contextual information
- Extraction of features from the data
- Making prediction based on the available data
- Understanding its faults and improvising itself to provide more accurate and useful information

Let us further look into each of the above mentioned step in detail to understand how they function and fit together.

2.1.1 Sensing

First and foremost step in any anticipatory system is to collect information about its users. The information is in most cases, quasi-continuos sensor reading from the device's various sensors and various other sources of contextual data available for the user. The nature of data varies on various factors such as the availability of sensors, portability of the devices and the actual requirement of the system.

With the advent of modern mobile devices we have a plethora of sensors encompassed in a single portable handheld device which the user is almost certain to carry around and use continuously throughout the day. This presents an opportunity for us to extract meaningful information about the user in his natural environment without interrupting or obstructing his day to day task and definitely without him deviating from his natural flow of work. Almost every smartphone these days has basic sensors such a GPS. accelerometer, light sensors among other and can usually connect to the internet. This provides us with a very rich source of contextual information which we can harness for gathering data about the user. This is not just restricted to smartphones but can be extended to various other platforms such as music players (iPod), tablets (iPad, Nexus 10, etc.), gaming consoles (XBOX, Wii, Playstation), home theatre systems, vehicle GPS navigations and many other components that we interact with almost on a daily basis, all of which provide some source of context which we can put to use.

When we perform sensing on an individual and collect information it helps us in understanding an individual alone and is usually referred to as *Personal Sensing* (Lane, Miluzzo, Lu, Peebles, Choudhury, and Campbell 2010). But with the recent growth in mobile internet networks, people are always connected and so are the devices to each other and to the web; this allows us to not only sense the information but also to store it onto a central location on the

Figure 2.1: List of sensors that are available on modern day smartphones. *(Apple iPhone 5S and LG Nexus 6 in the image).*

worldwide web. When sensing is performed on a larger scale i.e., where we are monitoring a group of individual or a certain community to identify patterns and emerging trends, this is termed as *Crowd-sensing* wherein individual devices collectively provide data to identify trends which are of a collective value to the entire community (Ganti, Ye, and Lei 2011). There are already several real world examples which use community based sensing in areas like environmental monitoring (Dutta, Aoki, Kumar, Mainwaring, Myers, Willett, and Woodruff 2009) and navigation (Mohan, Padmanabhan, and Ramjee 2008; Hull, Bychkovsky, Zhang, Chen, Goraczko, Miu, Shih, Balakrishnan, and Madden 2006). The only shortcoming of this technique is being unable to interact with the test users to take their feedback personally which is possible in the case of personal sensing.

Sensing even with all the vital information it provides is not without its owns issues and drawbacks. The availability of the data is very much platform dependent, with multiple devices and mobile operating systems, deployment is one of the key challenges along with the issues faced on each of the platforms. Additionally as it is with all areas of research which makes use of user's information, privacy of the user and of his information is a key concern. We will discuss this further in chapter 7.

2.1.2 Feature extraction and context association

The raw sensing data acquired from the sensors does not have much use, raw data contains a considerable amount of noise. We need to interpret the raw data to extract relevant features from it which is specific to the requirement of our application. For example we can have the raw data from the accelerometers but how do we understand from the accelerometer's x,y and

z values the activity user is performing, if the user is currently travelling how do we infer the mode of transport. How do we decide if the user can be interrupted at any given point in time (Ho and Intille 2005)? There are various platforms and statistical tools readily available for analysing and deriving vital information from each of the raw data types. However each of the contextual raw data types can be used to extract various different characteristics (Pejovic and Musolesi 2013). For example sound can be used to identify the present surroundings of the user however it can also be used to understand the physiological state of the user as well (Lu, Frauendorfer, Rabbi, Mast, Chittaranjan, Campbell, Gatica-Perez, and Choudhury 2012). So the feature extraction actually depends on what is the required end product and whats the role of the contextual information in making the end product a reality.

Additionally, there are other co-context's that can be fetched using the available contextual information. For example, having location of the place where the user would be in the next few hours enables us to fetch weather forecasts for that area, hence weather becomes a co-context. Similarly, if location itself is not directly available but we have cell tower information or WiFi information we can calculate the location from based on it.

2.1.3 Anticipation, improvement and self-learning

Understanding the features and extracting the information from the raw provides us with an opportunity to use this information for much more then understanding the present context of the user. By knowing what user's choices are we can make predictions for the future. For example, let us consider an example application which monitors location and accelerometer data using the user's mobile device. After a period of time through feature extraction the application would develop a user model according to which the user travels everyday at 9am to his office and uses the train as the means of transport. The application can then predict his travel schedule and provide train timings in the morning before he leaves his home. Additionally it can also notify the user if he is late by identifying deviation from his daily behaviour pattern.

From the extracted features various supervised learning models can be constructed which can associate user's activities and the related context. For example, in the travel prediction application, we can verify the user did take the train specified by the application or not based on which we can strengthen our prediction model. This when applied to the collective data of the community helps us build an effective model which can help us anticipate things which would be useful for any user in that community.

2.2 *WebSense* as an anticipatory system

WebSense along with the context sensing framework lays a foundation for various viable anticipatory system, even at present it partially fulfils role of an anticipatory system in the form of *WebSense*. To understand how the system works as an anticipatory systems we need to understand the various components which interact with each other and other components that can be linked with it to make it a complete anticipatory system.

The primary component of the system is to gather information for further processing. This task is performed using the context sensing framework, which provides all the context information

based on subscriptions. Other then context, user's application usage is also one of the most vital component of the system. This is extracted from the mobile's default API. The framework and the sensing process and strategies are explained in further detail in chapters 3 and 4.

Since the application falls in the category of crowd-sensing, the data is sent to a central data store on the web from the various devices of individuals. The application makes sure that the data is sent to the server in a power and money efficient manner. The server is responsible for storing the information securely and also acts as an authentication server. Essential features are filtered and context is associate with each other and the user. Further also tries to understand the extracted information and provide end points for the application to get prediction data and analytics information (See chapter 4).

As you will see in details in the chapters ahead, the application takes into consideration each of the aspect of the anticipatory system. There is much room for improvement in all of the components especially predication and self improving mechanism which will be further discussed in chapter 5.

Chapter 3

Android Context Sensing Framework

Context is a very primal component in all of the anticipatory systems. As one can conclude from figure 5.1, the contextual information that can be gathered using modern smartphones is a lot and more and more sensors are being added onto the devices with every new release. However, there are various mobile platforms and each consisting of its own development environment, tools and deployment stores making it very difficult to develop a framework which can function on multiple platforms. For the purpose of this project selected the Android development environment for various reasons. The platform this provides a much more flexible and open framework to extract contextual information. It also the most widely used mobile operating system and consists of devices of all price ranges hence providing a much more diverse target audience.

Although there have been many contextual frameworks in the past some of which are for Android as well; providing a relatively easy methodology to extract meaningful information (Novak, Carlson, and Jarzabek 2013; Rachuri, Musolesi, Mascolo, Rentfrow, Longworth, and Aucinas 2010), however the present requirement of the project is much simpler and there are some components that the frameworks presently do not monitor, for example, (Novak, Carlson, and Jarzabek 2013) does not monitor network, WiFi access points, (Rachuri, Musolesi, Mascolo, Rentfrow, Longworth, and Aucinas 2010) does not monitor user's schedule and phone profile. Also a framework at this point would provide more flexibility and access to the more recent API's which older frameworks do not cover. Although The context sensing framework designed for the purpose of this project but it is designed in a way to interface and connect to any application for monitoring purpose.

3.1 Contextual information types

There are various sensors that are available for the purpose of monitoring in the modern mobile devices and supported by the Android SDK. The following are the contextual information which were considered to be supported and available for subscription in the first version of the framework.

- Location (GPS or Network)
- Battery/Power source information

- Network connectivity status
- Network signals (Cell tower connectivity)
- WiFi access points and connectivity status
- User events
- Bluetooth status and nearby devices
- Telephony events
- Screen activity context
- User phone profile (Screen brightness, Volume settings & Vibration)

We will further discuss a few of the the above mentioned context's in details and how a scanning methodology was customised for it be both effective and efficient.

3.2 Framework's working architecture

In order to understand how the framework would actually function once integrated we divide the system's architecture into two components, first is the external end points of the framework which any application can interact with the second component is the actual internal component which handles subscriptions, monitoring, polling and content delivery.

3.2.1 End-points and external handling

As we described earlier the purpose of creating a new framework is to make sure information is provided in the simplest possible way and should be extensible. Also the framework has to be plug and play, there should be very minimal code that would have to be written on order to integrate it with an existing application and to perform what is required. The functionality of any existing application should not turn away from its original goal to focus on the integration of the framework and procuring data from it. The framework should just be a component integrated into the application providing a stream of required information and which can be controlled by the application.

The framework's architecture can be described as a subscription based model, wherein each of the components can be subscribed externally by the connecting application. In addition to that each subscription can be provided with additional parameters in order to tailor the subscription based on the needs of the application.

To begin monitoring any particular context one just has to call the *monitorContext* method from *ContextManager* object. The call then initiates a monitoring service for the context based the parameters that is being passed along with it.

As we can see in the function call, there are several parameters, let us look into them for more detailed understanding of the function's actual functioning.

```
public void monitorContext (
            ContextManagerServices mService,
            long minimumUpdateTime,
            long pollingTime,
            ArrayList<Integer> flags
    );
```

Snippet 1: Function definition of the monitor context.

- ***ContextManagerServices* Parameter -** *ContextManagerServices* is an enumeration defining various types of contexts that can be monitored by the framework, this specifies which context the application wants to monitor using the framework.
- ***minimumUpdateTime* Parameter -** This specifies the minimum time the application should wait before sending a forced broadcast about the context regardless if there is any change in the context or not. This is useful for application which want information about context at regular intervals regardless of the magnitude of the change.
- ***pollingTime* Parameter -** This parameter specifies the time interval after which the framework should poll the sensors or the operating system for information about the context. This is specifically for information which do not have system provided callbacks (Broadcast/Intent architecture) and require polling or in cases where power consumption for constant monitoring is too steep and is not feasible (e.g. GPS based location monitoring).
- ***flags* Parameter -** This is an extensible parameter which is essentially an array of flags which can be passed to the framework. This also provides a backward compatible way to extend the framework without changing the original function call definition. An example for the use of this parameter is for passing information about the source to use for location tracking to use while performing the polling task.

With the help of this single call, one could start monitoring any contextual information that is made available in the present version of the framework. In case of missing parameter related to the timing information, the framework uses a constant value for monitoring tasks which is customised for differently for each type of context.

To notify the application about contextual information the Android platform provides a very effective methodology to transfer information between the framework with the help of *Broadcast-Manager* which sends a message called *Broadcast* to all the *BroadcastReceiver*. The working of the methodology is very simple yet very effective. As figure 3.1 explains we can ask for an object to register itself to the *LocalBroadcastManager* to be notified in case it receives a broadcast with a particular name or tag which is referred to as *action* in Android.

The framework uses the same action string for all the notifications with more detailed information about the content being supplied inside the *Intent Bundle*.

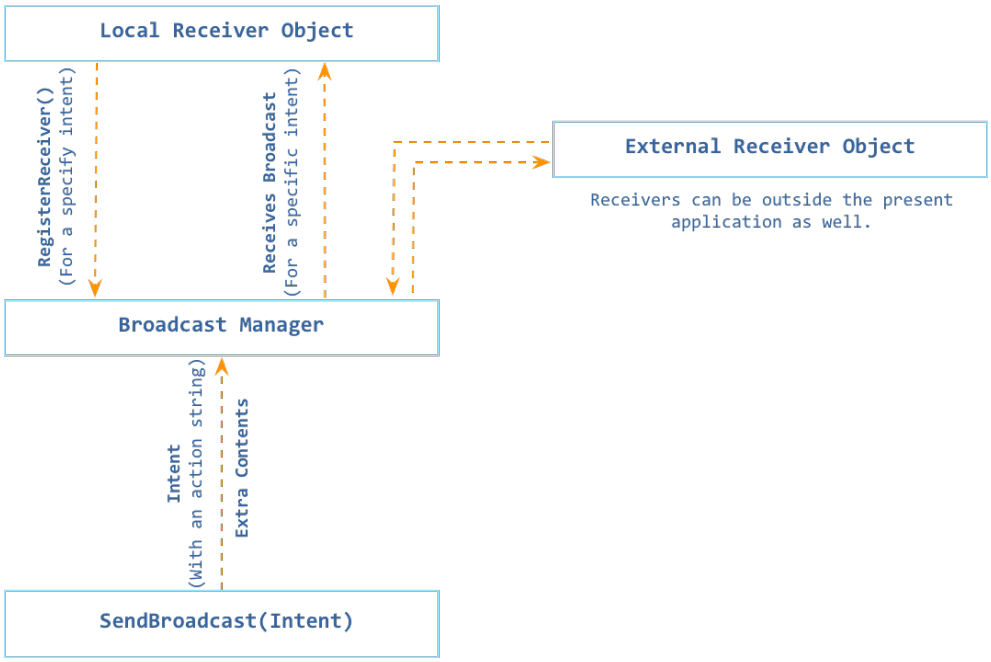


Figure 3.1: Architecture for the information broadcasting model.

The application using the framework would have to register a broadcast receiver just once to the*LocalBroadcastManager* for any and all contexts the framework is monitoring. When the framework has an update it would send a broadcast consisting of the following components:

- **Action -** All broadcasts from the application are associated with one action string '***CONTEXT_CHANGE_NOTIFY***' regardless what type of context it is.
- **Additional Information -** Other then the action string, broadcast also contains additional information in the form of a collection of key/value pairs. The framework sends various other values along with each of the broadcasts.
  - ***CONTEXT_TYPE* -** The type of context the broadcast contains.
  - ***Context Information Flag* -** This information is passed with the key named after the context type that is passed, hence varies depending on the content it contains. The value is a JSON (*JavaScript Object Notation*) formatted string containing information passed by the framework.

Finally, the application should be able to stop monitoring a context as easily as it begin monitoring the context. The context can be stopped from being monitored and hence the application would receive no further updates by simply calling

3.2.2 Internal working

The internal working of the framework constitutes on multiple components of the framework working in parallel and interacting with each other constantly. As described in section 3.2.1,

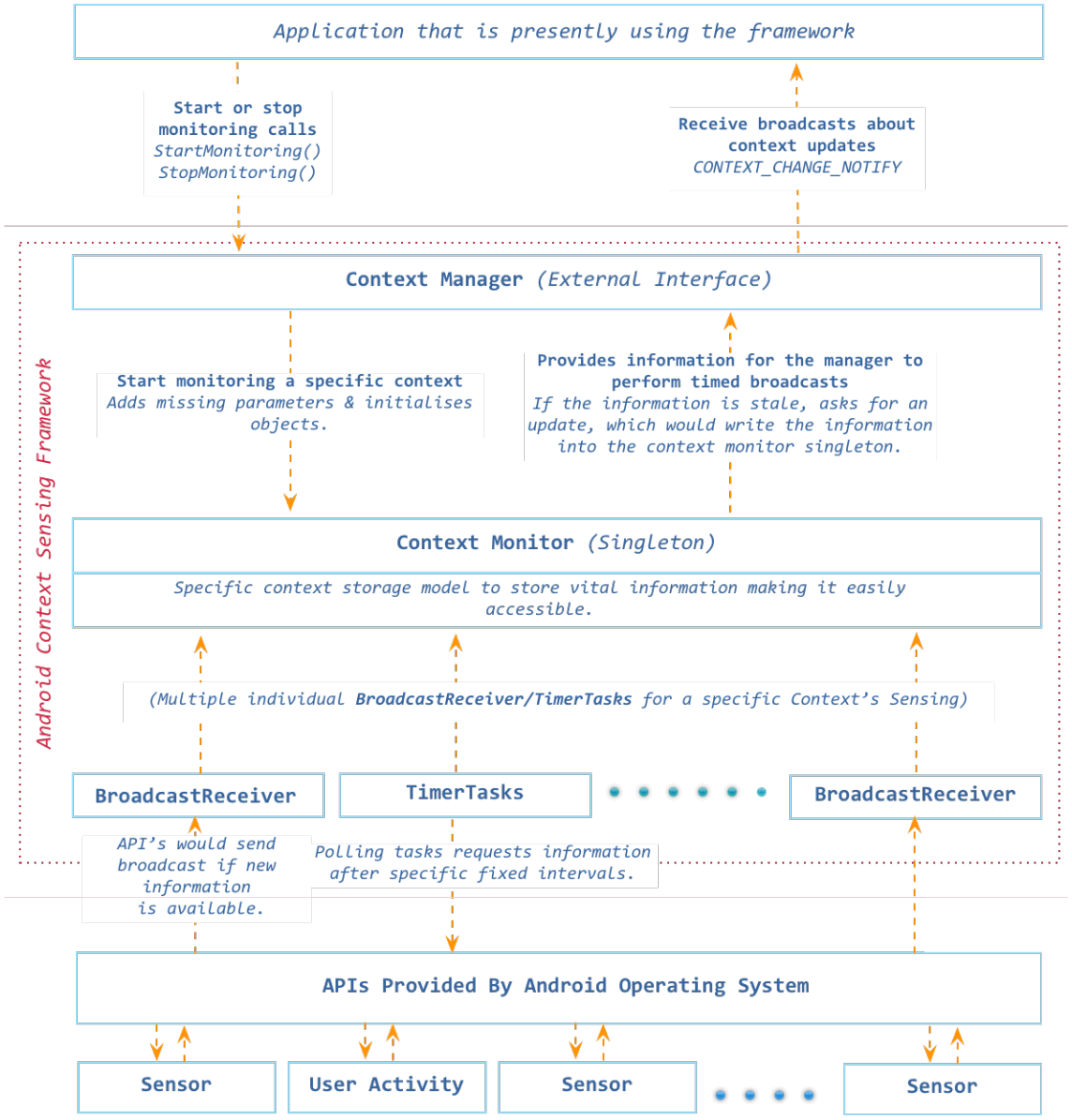


Figure 3.2: Architecture of the Context Sensing Framework.

```
public void stopMonitoringContext(
            ContextManagerServices mService
        );
```

Snippet 2: Function definition to stop monitor context.

ContextManager is the external interface with which applications trying to use the framework would have to interact with. The class is also responsible for delivering broadcast at specified *minimumUpdateTime* interval. However, the framework is build on a on a set of classes which ensure that latest information is always made available to *ContextManager* and is done is an efficient way.

A singleton class, *ContextMonitor* is primarily responsible for performing all the monitoring tasks at the specified time intervals and updating the values for the *ContextManager* to access and pass on further to the application. There are two kinds of context extraction process that the class has to deal with.

- **Monitoring Tasks -** Certain contexts have a broadcast and intent model provided by the operating system, in which case, a broadcast receiver for that specific action string is created in the singleton to receive information and update its variables accordingly and in case of a relevant change a broadcast is sent to the receivers. The subscription to receive the update is added and removed whenever the applications requests the framework to.

- **Polling Tasks -** Certain contexts need to be accessed using Android system API. These calls would have to be made at the specified time intervals to keep the information updated. For this purpose we use a timer task in *ContextMonitor* and is triggered when monitoring is requested. In case there is a relevant update in the information a broadcast is sent to all listeners about it.

  There are several advantages of using this development model, primarily this provides a single point of access to all the information that is required by the framework which provides easy of use and extensibility to the framework. Additionally, a single instance of the monitoring task ensures there are no multiple tasks being run by the framework for the same type of context which can be accidentally triggered in case the application using the framework starts to monitor same context at multiple places if the singleton was not in place. Although the framework provides a very resource effect method to gather information (see 6 for performance benchmark).

3.3 Understanding individual context monitoring

Each of the context's being monitored by framework have their own unique methodology in which they work. The uniqueness helps them to perform monitoring in a very effective and power efficient way. Let us look in details a two of the important context's working.

3.3.1 Location

Location is one of the most important contextual information that can be extracted by the framework. With modern devices which people carry everywhere it has become all the more important. The major concern with location monitoring is the power consumption factor when using GPS (Ben Abdesslem, Phillips, and Henderson 2009), so continuously monitoring of GPS is never recommended on mobile devices. Also, time it requires to switch on the GPS sensor.

Getting a triangulation from the satellites can also be a time consuming task sometimes. Modern devices provide network based location services which require both network connectivity and internet to provide location. This process is battery efficient but accuracy is usually questionable.

The framework provides the application option to opt for either of the technique in addition to the ability to specify the polling interval.

Android operating system stores the last known position[1] which is updated when any of the application tries to fetch location using GPS or network provider. So once every few minutes (defined by the provided polling time) the device first looks for location that is stored using the *getLastKnownLocation* method. Then the framework analyses if the location that is being provided by the system is a better location then the one it already has and if it meets the criteria (see figure 3.3), if the criteria is met this location is stored and broadcasted. If not, the system would trigger the specified location provider (GPS or Network) to provide it with a new location, the fetched location is then broadcasted and stored.

3.3.2 Network information

Mobile phones are almost always connected to a cellular network, when the user moves the network connection also changes. The systems decides which of the towers providing signal to connect to out of the various ones that are available. This information can be very useful in terms of context. For example with this information one can map the network density and signal strength of the area. There are several components that constitute of network information and there is not a single method to monitor this.

- **Signal Strength -** Signal Strength constantly changes and can be monitored by a system provided broadcast receiver. However the frequency of the change is very high, hence broadcast is done only when there is a significant change in the signal levels.
- **Cell Towers Nearby -** Although the device usually connects to a single cell tower but there are various other towers which broadcast their presence and the device picks it up along with other information such as their signal strength. This information is also provided through a broadcast receiver by the system.
- **Data Connectivity -** The networks may or may not be providing data connectivity to the device or the user might have turned off the data connectivity. This information needs to be polled constantly as the broadcasts are not reliable and do not provide internet connectivity information. The framework frequently checks for updates about connectivity information and notifies if there is any change.

3.4 Shortcomings and possible improvements

The framework provides a solid foundation with features that make it extensible and might provide useful to various kinds of application and research work on the Android platforms. There are several areas which should can be optimised and improved further in the future versions.

[1] http://developer.android.com/reference/android/location/LocationManager.html#getLastKnownLocation

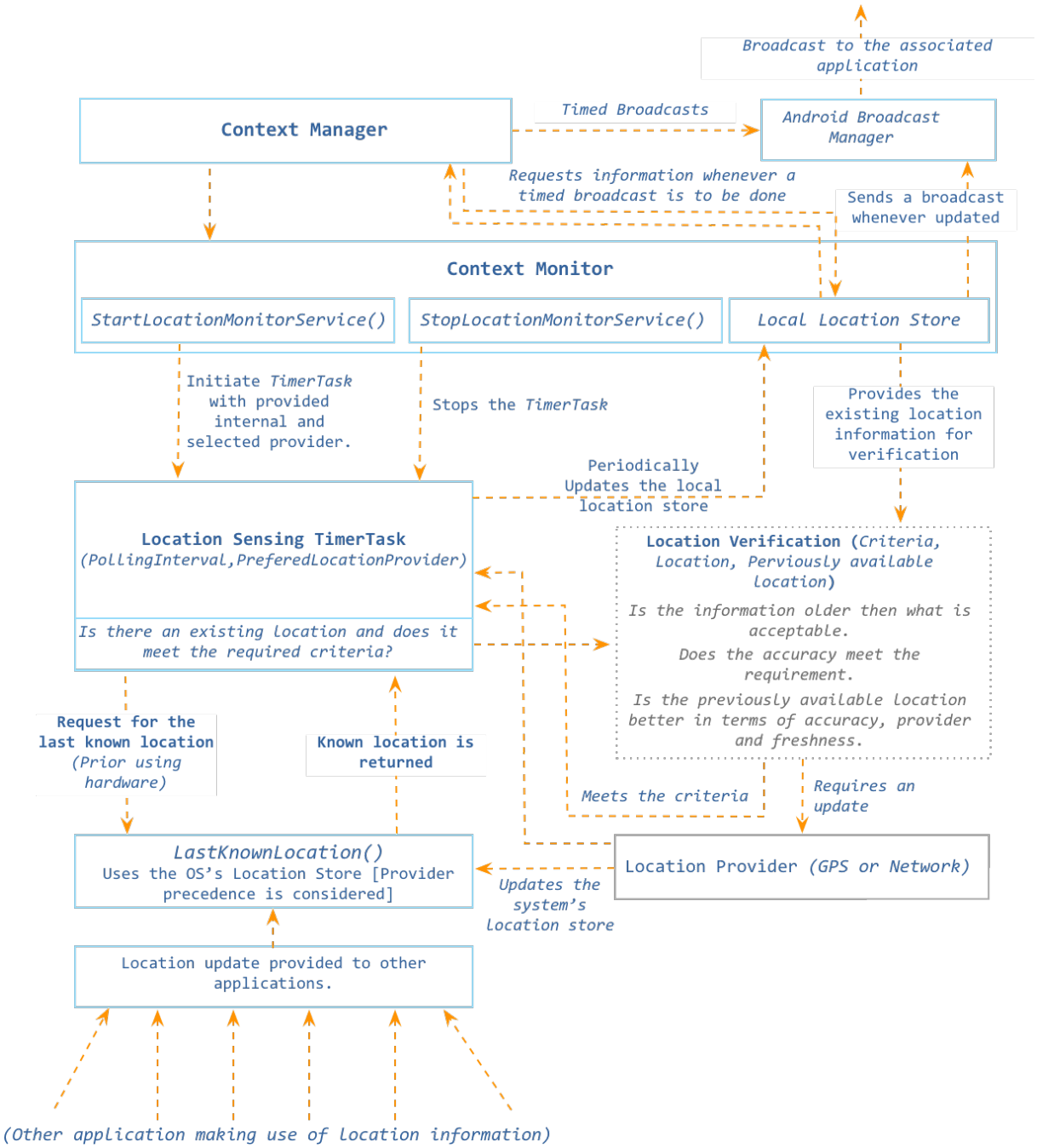


Figure 3.3: Process for efficient monitoring of location.

Firstly, the framework can be made more robust and reliable. Currently it has been designed for devices running Android version 4.0 and 4.4, backward compatibility would certainly make the framework useable on a wide range of devices providing more diversity to future studies.

Although android is at present one of the most popular operating system in the mobile market, there are a considerable number of the mobile users who use other popular operating systems like iOS and Windows platform, an extension of the sensing framework with similar functionality (within the bounds of the operating system) on other platforms would be certainly helpful in conducting studies with a broader group of audience then just people using android.

Additional context's like accelerometer, proximity, ambient light and application monitoring (currently implemented in an external build over the framework) are presently missing in the framework and would be valuable additions to it.

The sensor information gathering technique is very basic in nature and lacks adaptive qualities like in more complex libraries EmotionSense (Rachuri, Musolesi, Mascolo, Rentfrow, Longworth, and Aucinas 2010), which allows the sensing to be done in an adaptive manner instead on a fixed interval. For example movement of the user should be tracked based on the movement he has done during the previous time interval, if the location change between two interval is considerably less or none at all, the next sensing should take place an increased time interval then usual or if the change is considerable the sensing time interval should be reduced. Additionally, it should take into consideration the battery usage of the sensor and the current state of the device, i.e. sensing intervals should adapt itself based on the battery life of the device to improve overall performance of the framework.

Given enough time the framework can be expanded into a much more robust, functionally rich and highly customisable component which can be used in various kinds of studies and real world application.

Chapter 4

User's smartphone usage data collection application

The most essential goal of this project is to devise a technique to gather information about how a real world user interacts with his mobile device on a day to day basis in his natural and unmoderated environment. The context sensing framework provides a very robust foundation using which we can gather information about the context in which the user is using the mobile device. But at present the framework does not consist of any possible way to monitor and gather information about mobile phone interaction of the user both in terms of web and application usage.

WebSense is an application for the Android platform developed for the very purpose of collection of user application usage data collection along with the related contextual information. The application is integrated with the context sensing framework described in chapter 3 which it uses to extract contextual information of the user.

There were two aspects of the application which were taken into consideration while designing the architecture of the application. First was the actual development of the monitoring functionality to provide an efficient way that was to be used to monitor how the user interacts with the device in a ubiquitous way and eventually send it to a central data repository on the web for collective analysis. Second aspect was the utility value it should provide to the user which is essential for crowd-sensing application as apps in the crowd-sensing domain should be lucrative enough for people to install even if they are not interested in the research value of the application (see section 7.1 for deployment challenges). In the following sections we look into each of the aspects and the various components which collectively provide a skeletal over which the application functions.

4.1 Application usage demographics collection

Android is a very open platform in terms of providing access to data sources when we compare it with the other most popular operating system for mobile platforms such as Apple's iOS (Apple Inc. 2013). This allows us to develop application which can monitor user interaction in detail.

The most essential criteria of any monitoring application is to maintain a near 100% uptime,

which means that the app should always be running in the background and performing the necessary operations. Implementing a front end interface provides user an option to terminate the process very easily by accident or intent hence denying the change of gather information. However a background service on the other hand is more complicated to be terminated and also ensures that no hinderance is caused to the user in performing his day to day mobile phone usage. Hence the architecture of the app consists of several of background services each with their own set of responsibilities. This, along with a design to ensure that the monitoring service is switched on in case of device restart, accidental termination by the operating system or in case of an internal malfunction causing the termination of the application provides the system to operate at a near 100% uptime (see chapter 6 for detailed evaluation).

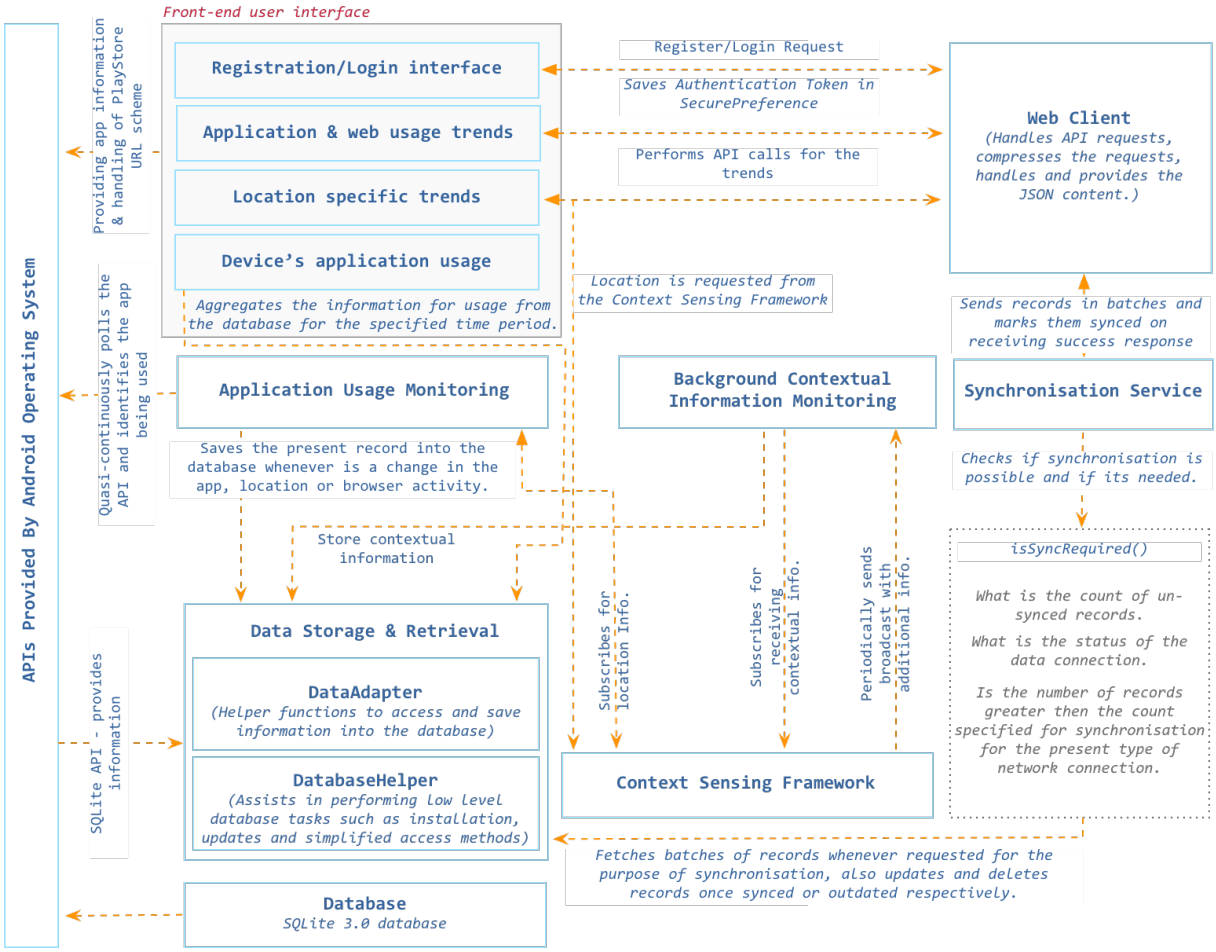


Figure 4.1: Architecture of *WebSense* Android Application.

Let us look into the various background services along with their role in the overall architecture of the system.

4.1.1 Application usage monitor

The primal responsibility of the app is to monitor the user's application usage continuously and with certain level of dependability. This task is performed by the service *AppUsageMonitor*. The service is a self-reviving service so in case it is terminated either by the operating system or due to a malfunction it revives itself within a matter of seconds, ensuring an almost 100% uptime.

Since Android does not provide a broadcast or any sort of delegation methodology to know when the application is switched, the only possible approach to do this at present is a quasi-continuous monitoring of the device's application stack. An interval of 4-second was chosen between each scan since research has indicated that user's attention span switches between the device and the environment somewhere within this duration (Oulasvirta, Tamminen, Roto, and Kuorelahti 2005), hence we can assume that the user would switch applications as well at around the same frequency. Android provides access to the list of currently active processes through the *ActivityManager* class. The class not only provides information about the running tasks but also provides a mechanism to access the order of the application's screen presence. With this information we can identify which application is currently running and being interacted with on the device by the user.

Android also equips its developers with intents which one can listen to for detecting user's action of turning the device's screen on and off. This allows us to stop monitoring user's activity tasks when the phone is turned off hence saving crucial resources.

The application uses SQLite database[1] to store the monitoring information. This includes package name of the application, start time, end time (Both in UTC timestamp format), duration of the run, location it was used at (subscribes to the framework, listens and stores location information constantly) along with the day of the hour the application was used at. Additionally, a flag marking the synchronisation state of the record is also maintained.

Whenever an application switch is performed the data is written into the database hence minimising the chances of data lose incase of a malfunction or termination. The information is also saved in the scenario when location update is received while an application is running, hence splitting a continuous session of the application based on the location. This is particularly useful in case of navigational applications as the user would keep it open over an extended period of time while his location changes constantly.

If the application is detected to be the default browser of the device, the app also gathers information about the page user is presently viewing. Although, there is no direct way to perform this, android provides an API to access browser history[2], the most recent record in history would be the page the user is viewing currently. Whenever user changes the URL a new record is created and hence the time spent on a particular URL is also recorded.

4.1.2 Contextual information monitoring

ContextBackgroundMonitor class is also a background service which is responsible for registering and monitoring the contextual information. Presently it monitors all the contextual data provided by the framework (see section 3.1). It is responsible is to make sure that as soon as the

[1] self-contained, transactional SQL database engine https://sqlite.org/

[2] Browser Data Provider - http://developer.android.com/reference/android/provider/Browser.html

service is initiated it subscribes to all the available contextual information provider and whenever there is any new information made available by the framework it saves it in the database. Since we require information about location for mapping the information geographically and also have to provide user with information in relation to his location, all the context information is stored along with the latest location information attached to it.

The class is initiated as soon as the monitoring begins and when the class is destroyed it releases all the receivers and initiates the stop procedure in the framework (see section 3.2.1) hence stopping monitoring process altogether.

4.1.3 Web synchronisation service

The context sensing and the application usage monitoring generates a considerable amount content in terms of memory. This can directly be used for the purposes of local mining (see section 4.2.2) but this being a crowd-sensing application, we need the information at a central location to further analyse information of the user-base as a collective. For this purpose, the information that is in the local datastore has to be constantly sent to the remote server on the web.

The class *SyncManager* performs all of the synchronisation of the device with the web by uploading the information in the most efficient way possible. After a certain interval of time the class triggers a method to check for a possibility for a need of uploading of the information. Records which have not been synced in the database is the first criteria the method checks for. To avoid repetitive reads to count un-synced records the monitoring services automatically increments the count whenever a new record is added and stores it on a preference file, this file is read by the synchronisation process. If the criteria for data upload is met the information is uploaded in chunks of fixed number of records in the form of a JSON string and once the data is successfully uploaded the records that were sent are marked as synced. The synchronising process uses the end points provided by the web application and sends the data after compressing it using GZIP [3]. The records are sent with an authentication key to both validate and associate the records, this key is obtained during the login process (see section 4.2.1 and 4.3.1).

The service is also responsible for periodically clearing old data which has been synced already and are beyond the time-period which is relevant to the application (30 days in this case).

4.2 User interface & Providing utility value to the user

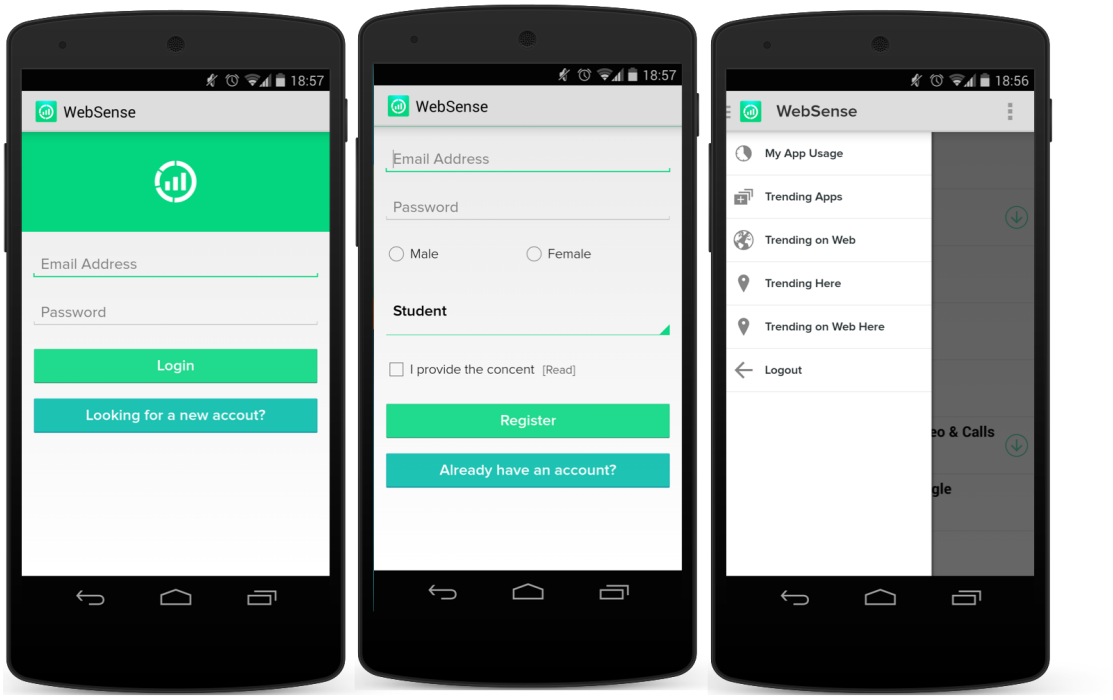
The monitoring and reporting aspect of the project could have been done without actually a front end with everything entirely handled by background services but the primary test subjects of this application are not a selective group of people, like any crowd-sensing application the future research based on this application tends to make this available to as many users as possible, not only the people who are interested in the research aspect but common public with Android devices. As we will see later on in chapter 7 one of main issue when dealing with crowd-sensing application is that it is bottle-necked by the fact that the entire research, application's widespread usage and the amount of data gathered depends solely on the decision of the user to install and keep the application on his phone. So to ensure widespread participation and retention of the

[3]GZIP File format specification -http://tools.ietf.org/html/rfc1952

application we need to provide the user utility value through the frond end of the application, so regardless if the user is concerned with the study or not he would still have a reason to install the application.

The user interface of the application which runs atop the data gathering services provides Information and feedback to the user and also helps in gathering certain user's personal information. The application is also integrated with a crash reporting framework called ACRA [4] which provides the crashes and issues faced by the users to the web server, helping us to improve the application's performance and making it more stable.

4.2.1 Login/Registration interface



(i) Login interface (i) Registration interface (i) Navigation drawer interface

Figure 4.2: Initial screens of *WebSense* app related to authentication and basic navigation

The login/registration interface is the initial screen of the application where in the user provides the below mentioned information if he is registering or just logs into the application if he has already registered.

- Email Address - Allows us to have the means to contact the user in the future.
- Password - For authentication purposes in case the user logs out or wants to login on another device.

[4] Application Crash report for Android - https://github.com/ACRA/acra/

- Gender - Can be used as a metric to perform future studies.
- Employment Status - Additional parameter which can be used in future studies.
- Consent to allow monitoring - Along with a link to the End user's license agreement.

The application allows multiple sessions using the same credentials on multiple devices. An option to log out of the application is also provided inside the application, in case the user wants to do something privately or wants to stop monitoring completely.

4.2.2 Application usage information

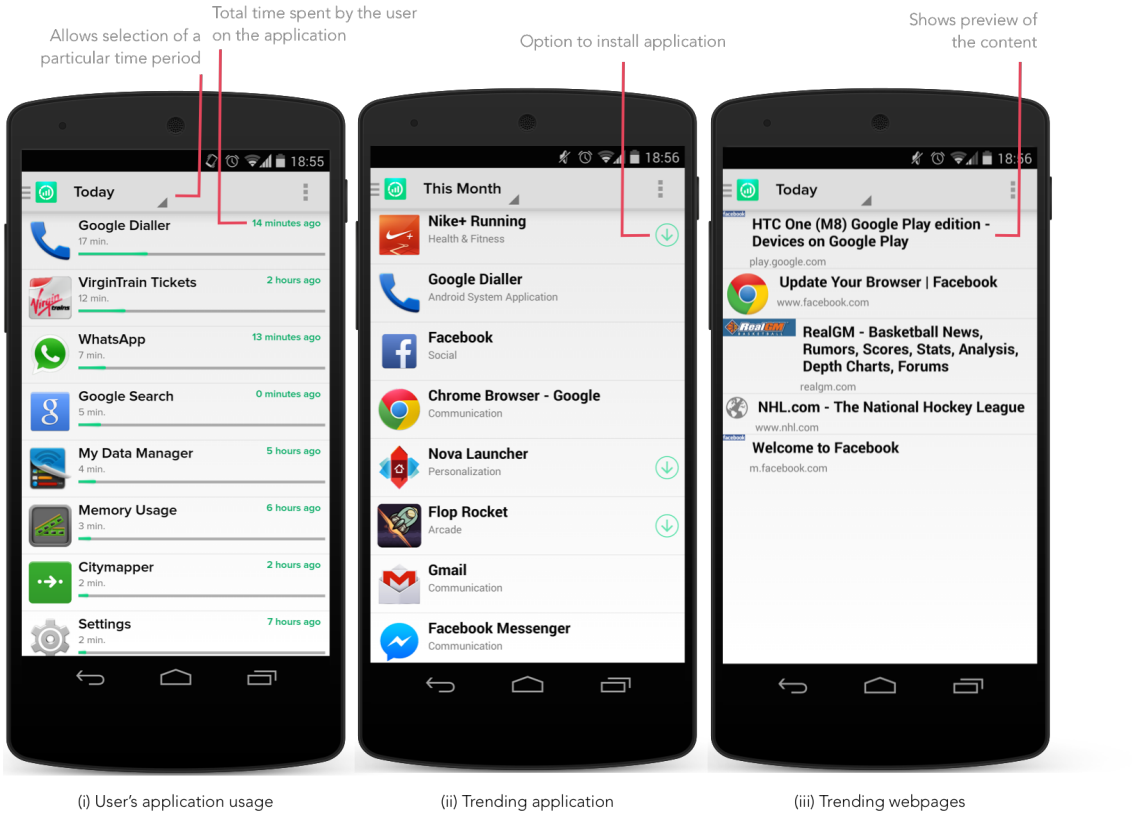


Figure 4.3: Content display screens of *WebSense* app providing trends and usage information.

Application usage of the user is being constantly monitored by the application in the background, this interface provides a view by accumulating the total usage over a certain period of time. The interface provides information about the total usage in a very visual way allowing users to compare the various application. An option to filter the information to a particular duration (day, week and month) is also available.

4.2.3 Application and web usage trends

This provides similar as the previous screen but instead of providing information for user's application usage, it provides aggregated information about the trends throughout the user-base of the application. It shows the trends of application usage based on the data accumulated over the period of time on the web server over a period of time. The web API has an end point which provides the information for the required duration of time (day, week or month).

The application's information is also provided along side, hence icon and the name of the application is shown along with an option to install the application if it is not already installed. This acts as a recommendation system for the user to try new application based on popularity.

Along with the application usage trends the app also shows information about the web usage trends, showing information along with some content and images from the websites popular among the masses in the given duration.

4.2.4 Localised web and app usage trends

The information about the app and web is can also be restricted to the present geographical area. The app has the capability to present the user information about what is popular in the user's current area. This can be very helpful for example, if the user is opens the application at a train station the app would provide information about the apps and websites people use there, which are most likely to be the ones providing information about the timings of the train.

The geographical radius is decided on the server the location is provided by the context sensing framework which then is passed along with the web request. The process of handling this request is discussed further in detail in section 4.3.2.

4.3 Web API and feature extraction

As we have seen the application provides a robust and lucrative way for the users to provide their application usage information, however in order to achieve this at a larger scale and not just for a single user we need a web server to handle storing of information, extract features and finally process it to provide meaningful information for the application to display and even perform predictive tasks at a later stage. For this purpose a web application was build and deployed on the internet. The application is build using Node.js [5] which provides a very reliable framework to build high performance web API's, dealing with an increasing number of web requests (Tilkov and Vinoski 2010) which is bound to scale over time. The datastore for the application is a MongoDB[6] database which provides various functionalities to process and extract information efficiently. It also provides lot of features to improve handling of high request rates and take care of the needs for scalable systems.

The web app is essentially a collection of RESTful API end points with JSON data format to store, process and provide information, all of whom incorporate GZIP compression in their

[5]Even-driven JavaScript backend http://nodejs.org/

[6]Non-relational, Document based database https://www.mongodb.org/

requests and responses to decrease bandwidth being used. It also constitutes of a analytics dashboard which showcases various metrics from the server. Let us briefly go through the available API endpoints and their internal working.

4.3.1 Storing information

The application stores various types of information. The first possible request an app can make is to register or authenticate the user. The information about the user is stored in the *user* collection. The user document is designed to handle multiple devices and multiple sessions running simultaneously. The document model also stores other information that is sent by the app when the user registered which can be used as a research parameter for future studies.

When authenticated the service checks for the user record along with the device information and either creating or updating the device information that is provided with the login information and returning a unique authentication key which can be used for all further requests.

Storing of web and app usage information is more complex, when the service is called the first thing the server checks is if it is a valid service in terms of the authentication key that is being supplied. If it is, then it starts processing the information that is provided. The user is almost instantly provided an OK status message while the processing is going on.

It collects all the package names that were sent in the request and checks the existing collection of application information for their records, the ones which are new are then sent to a different service call which asynchronously checks each of the package name and fetches information about the application from the app store by scrapping the web page for information. The location information is also corrected and stored in the required format to perform geospatial queries later on.

The process for storing context information is also identical, except there is not much processing at present and once the information is cleaned and converted into required format it is directly pushed into the *context* collection.

4.3.2 Extracting meaningful information and patterns

The web app is also responsible to identify and provide patterns identified from the stored information to the application on the device about application and web usage. The information is extracted from the database using the MapReduce technique which MongoDb supports. The details of the technique are explained in the Figure 4.4.

The process for extraction of information usually involves aggregating the app usage for each package, however,this can also accommodate filtering based on parameters such as like duration and location information. Location filtering uses the geospatial functions that is provided by mongoDB to reduce the records to a particular area only. The radius along with things such as packages to ignore can all be configured in the config file for the server allowing dynamic changes to the server. The diagram above shows in depth the working of the functionality.

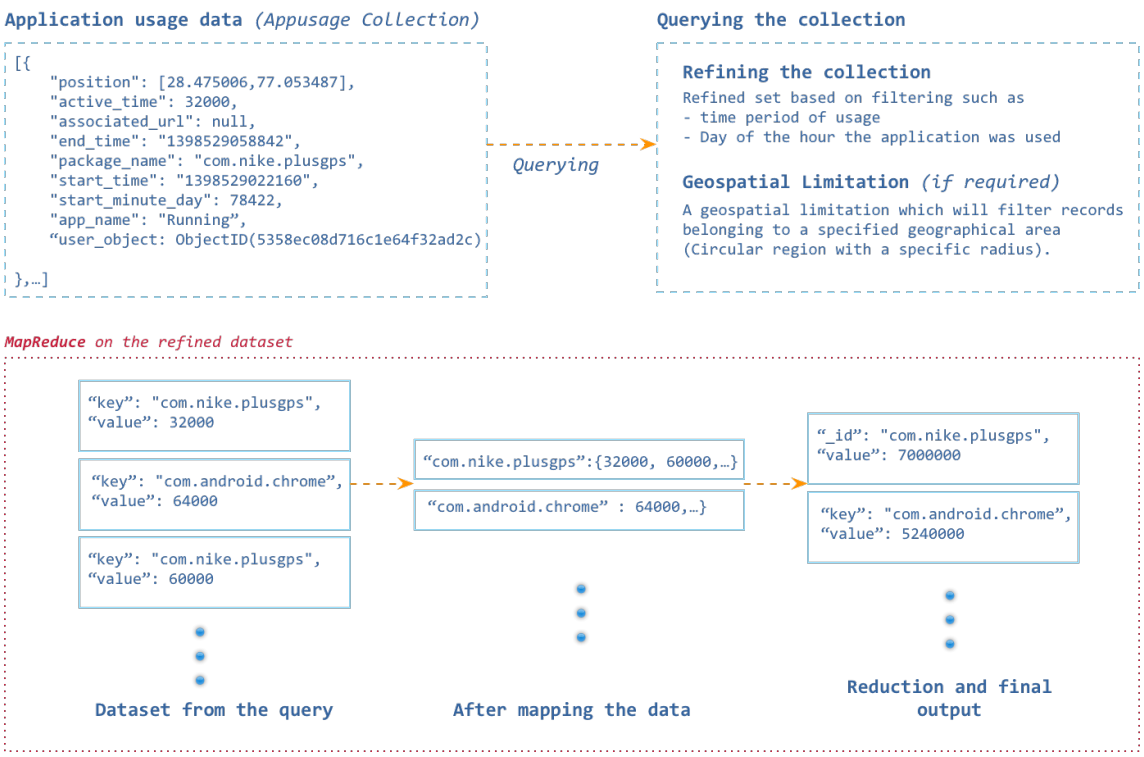


Figure 4.4: Applying MapReduce on the app usage data.

4.4 Shortcomings and possible improvements

The application and the framework were both developed over a period of 1.5 months. The purpose of this project was to lay an initial foundation for a robust data gathering application along with a framework which would be re-useable component for context extraction both of which can be re-shaped and used for studies beyond the scope of the present project. This being said, there are various improvements that can be done in the existing application and its web counterpart.

4.4.1 Android app

The android application has a few shortcomings in this first release which can be improved in the future releases along with addition of new features.

- Firstly, any improvements that were mentioned in relation to the framework (see section 3.4) would effect the overall performance of the app, both in terms of resource consumption and the amount of data that can be gathered using the application.
- The application at present fails in monitoring web usage if they not performed on the default browser which is accessible by the Browser data provider (*Chrome* on Android 4.0 and above) in the Android SDK. This can effect the overall demographics of the web usage data being collected as interest of the users using other applications to browse internet would not be registered by the application.

- The application also fails at monitoring tasks which can perform their functionality in the background while user is either using some other application or has switched off the device's screen for example applications such as music players and radios can play audio for the user while running in the background without having the front-end open.
- The application presently provides analytics and trends information which is very generic which can be customised and converted to a personalised recommendation system among various other possible improvements that can be done to provide more value to the user as an utility application (see section 5.2).

4.4.2 Web component

The web component can be improved considerably by adding more features in addition to improving overall performance and scalability of the system.

- The web component at present does not deal much with the context information, it simply makes use of app usage and location information to provide various results. Future features can make use of all the available information both for research purposes and also for implementing in the application.
- The web component's security can be improved by using *Bcrypt*[7] to store and encrypt private information. Additionally, more precautions can be taken by disassociating the contextual data from the user (see section 7.3).
- The database connection at present picks a connection from the connection pool of the driver, which is a reusable pool of already active connection helping in reducing frequency of opening and closing of connections (MongoLab 2013). There are several things that can be done to improve the speed network transactions.
  - Sharding[8] maybe applied and to the MongoDB instance to improve the performance of the application and handle increase in the data needs.
  - At present whenever there is an API call the web service triggers the MongoDB driver along with its customised query which reads data from the database, process it and returns it back to the service to be delivered further to the device. Considering the number of records being created by the devices, once this application has become widespread this process would start taking more and more time to process the request. A caching layer which would store the information in memory instead of reading raw information from the database and processing it again can improve the performance of the queries. Technologies which are meant to handle rapidly changing content in the memory such as Redis[9] can be used for this purpose in the future.

[7]Bcrypt: Bowlfish file encryption http://bcrypt.sourceforge.net/
[8]MongoDB Sharding - http://docs.mongodb.org/manual/sharding/
[9]Redis: An Open source key-value datastore - http://redis.io/

– Use of MapReduce technique to perform various complex functions in the application benefits the overall performance, but at present each time the service is called the MapReduce process is performed. This can be avoided by using incremental MapReduce[10] which automatically reduces the information when new information is added incrementally hence reducing the load on the server by a considerable amount. Future implementations can also look into a Hadoop based MapReduce implementation on the existing MongoDB instance as it has shown promising results (Dede, Govindaraju, Gunter, Canon, and Ramakrishnan 2013).

[10]MongoDB: Incremental MapReduce - http://docs.mongodb.org/manual/tutorial/perform-incremental-map-reduce/

Chapter 5

Putting the collected information to use

The primary purpose of this stage of the experiment is to design an effective method to gather information from the user on a large scale along with an effective delivery method to do so. The information being gathered is presently stored and and basic feature extraction techniques are being used to provide analytics and trends within the community and in a certain geographical area. However considering the diversity and richness of the data that is and will be available as a result of this application can provide useful input to serval interesting areas of research and help develop various practical applications.

Let us look at a few possible novel approaches to put the information to use in the future.

5.1 Intuitive Application Launcher

A very simple application on the lines of the FALCON system (Yan, Chu, Ganesan, Kansal, and Liu 2012) which we are already working on would be to build an intuitive launcher for Android devices based on the predictions that would be derived from the gathered information. However, rather then actually launching the application we can provide a much more simplified approach by not pushing the user to make a choice and saving considerable amount of resources.

Mobile devices usually turn their screens off if there is no activity from the user. Whenever user has to use the device the screen has to switched on and the lock has to be opened. In the event of the screen being switched on, the application which would be integrated with the device's lock screen would present the user with options to launch certain applications. The list of application for the user to choose from would be modified based on time and location. This kind of predictive computing can be very helpful for the user and can prove to make the user's smartphone usage more efficient.

We already have contextual information about the user along with his application usage pattern which we can associate to extract a possible list of application he might want to use. An effective development method with the processing divided between the device and the server would be required to make sure the list can be provided without internet connectivity.

Additionally, the application can learn and improvise itself based on the choices the user makes on the launch screen, i.e., whether or not user selects an application from the list that was provided hence making the application better at predicting application selection over time.

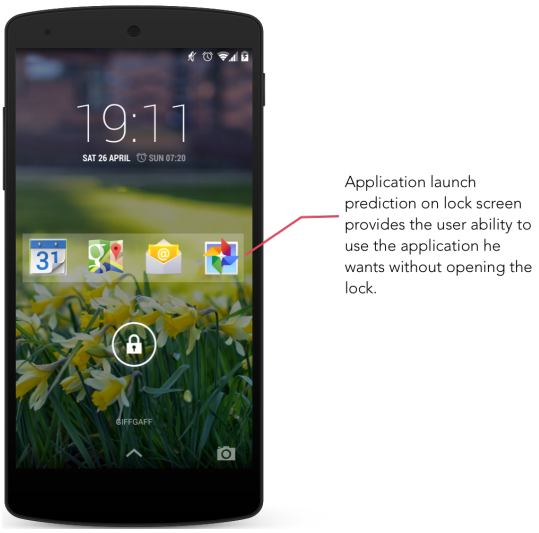


Figure 5.1: An interface prototype for an intuitive application launcher on the lock screen.

5.2 Recommendation system

The present version of the application provides recommendation in the form of *What's Trending* using the information that is available about the user's app usage and the location context. The recommendations however are very generic in nature, i.e., recommendations on the basis of the user-base or geographical area. There can be several improvements to the existing features. The system could recommend users what is trending based on their mobile usage pattern, for example if a person uses a certain application for ordering food frequently and there is another application which is popular in the geographical region the application can make a suggestion to him to try out the other application. The prediction can also be influenced by other information that we have collected like gender and user's employment type.

A similar feature can also be developed for the web usage. A recommended reading list can be compiled based on the user's reading habits and the present trends. To achieve this in practicality would require natural language processing to analyse the content of the webpages to understand the relevance of the link. The application would have to understand the user's web usage pattern and derive a web profile for the user and improving it over a period of time based on user's choices and web usage. The content which is recommended is already popular among people the possibility of a favourable recommendation is more likely. This would be similar to the idea presented in (Balabanović 1997) but would be for mobile devices; which would help the system to learn and adapt faster.

5.3 Content and application pre-caching

Information gathered about content being accessed on the web or application being launched alongside the location context provides us an insight about what is popular in a particular geographical region. This as discussed earlier can be personalised to provide what user tends to

view or use at a particular location frequently. Application can be launched and kept running in the background as to reduce the wait time for the user when the application is launched and possibly loading itself into the memory or maybe loading content from the web.

In relation to the web, this information can be used in a very different scenario. With the knowledge about localised trends on the web an interface can be created which can be made accessible externally to provide possible websites people might use at a particular location. This information can then be linked with the local cellular network or WiFi access points and the content can be cached on the network itself provided improved loading time for the users of the network hence improving the overall user experience. This can be further improvised further by anticipating and then caching possible future navigation from the current website reducing the loading time of requests even further beyond what is achievable using the trends information (Cheluvaraju, Kousik, and Rao 2011).

Chapter 6

Performance evaluation

In this section we presents the various statistics, results and observations obtained by running *WebSense* on 3 different android devices for 2 consecutive days. The information was stored on the server and processed and provided back to the device showcase on the user interface. We look into the various parameters that govern the performance of the application and the server component and then we also try to analyse the results and try to find the possible reasons and improvements that can be done based on it.

6.1 Application & framework performance

The performance of an application the framework is benchmarked based on various factors. The hardware itself plays a very vital role in the process. In order to better understand the performance on a much broader scale we use 3 different devices for 2 days in an experiment conducted over 6 days. During this duration the devices were constantly taken out and used as personal devices providing us results for normal mobile usage.

The devices that were chosen were, LG Nexus 5, LG Nexus 4 and Samsung Galaxy S5. All of them running Android version 4.4.2. The specifications of the devices are as follows.

- **LG Nexus 4**
  - 1.5 GHz quad-core Snapdragon S4 Pro processor
  - 2 GB RAM
  - 2100 mAh battery
  - 16 GB Internal storage
- **LG Nexus 5**
  - 2.26 GHz quad-core Snapdragon 800 processor
  - 2 GB RAM
  - 2300 mAh battery
  - 16 GB Internal storage

- **Samsung Galaxy S4**
  - Snapdragon 600 (quad-core 1.9 GHz Krait 300 CPU & Adreno 320 GPU)
  - 2 GB RAM
  - 3600 mAh battery
  - 32 GB Internal storage

Having multiple devices helps us in averaging out the performance and helps in detecting platform specific anomalies in the system. Firstly let us look into the stability of the application, using ACRA we are able to analyse crashes and issues faced on the device without being connected to the development machine, it catches crashes that happen in the background thread as well. Over the duration of 6 days the server received 1 crash report which when analysed and attributed to concurrent writes being performed onto the preferences causing a deadlock on a rare occasion. Apart from the crash-logs, we can also analyse the stability of the application by analysing the uptime in the settings application present in the OS which allows us to see the uptime of background services. On the Nexus 5 device each of the services had an uptime of over 90 hours, well over the experiment period. Similar observations were made on Nexus 4 and S5 as well. This verifies the application and the framework can provide an near 100% uptime in monitoring user's actions.

Let us look into memory foot-print of the application. The app constantly keeps alive 3 services to monitor and synchronise information, even if the operating system destroys the services in case of low memory (Google 2014a) the service's self reviving cycle brings it at back as soon as possible (See appendix 3 for revival time). The application does not require a front end for monitoring so it there is no constant front-end memory footprint. Using Android's DDMS (Google 2014b) and an additional app called Memory Usage[1], the memory usage of the application was analysed to vary between 16MB to 24MB. The peak in the usage when the application has the front end active and is performing web requests to fetch data. It has a similar footprint across all the devices.

The next crucial aspect is power consumption; the power consumption various with the various settings of the application, for the present configuration (See appendix C). The application consumed 4%, 6% and 7% battery on Nexus 5, Nexus 4 and Galaxy S4 during the experimental cycle.

In terms of data bandwidth usage, the application has shown to have a very low requirement. The web requests being made are all GZIP compression enabled which allows to reduce the call footprint drastically for the provided batch. In addition to save the information the system makes use of WiFi over data connections, which too is governed by the configuration factor (See appendix C).

The application transferred over 1350 KB of data over WiFi on an average and only 120 KB on cellular data. However, this statistics is highly variant as it depends on availability of WiFi and the mobility of the user.

Overall, from the evaluation of the application we have learnt of a few new issues in the application, areas that we can look into like reduction of the memory footprint. In addition to this

[1]https://play.google.com/store/apps/details?id=mem.usage

more detailed test related to power consumption need to be conducted on various configurations in order to tweak and improve the overall performance of the application.

6.2 Web application performance

The web application is built on Node.js with MongoDB as its database. The server is an Amazon EC2 instance with 0.613GB of RAM and other similar variable low end specs. We analyse the performance of requests which constitutes of mechanisms such as Geospatial requests and MapReduce in place; processing of the data over a set of 8580 records of application usage and 7362 records for context information (integrated at a later stage).

Over its short duration of being operational the web component has already scrapped information about 78 android apps and 33 websites consisting of image URLs, ratings, description, category , etc.

Over the duration of the experiment on three devices combined there were 2477 records created on the server with an astonishing 33.28 hours of phone activity, resulting in an average of 45 seconds per activity (Please consider that change in contextual information is considered as a different activity).

API end point	**Response Time**
/app/trends/monthly	199ms
/app/trends/weekly	113ms
/app/trends/daily	45ms
/web/trends/monthly	27ms
/web/trends/weekly	22ms
/web/trends/daily	11ms
/app/nearby/monthly	109ms
/app/nearby/weekly	75ms
/app/nearby/daily	35ms

Table 6.1: Response time for API end points.

Table 6.1 highlights the request response time of the various request that can be made. It is vital to note that the time required to process request relies on the number of records that it has to process and the network speed. So when new users are added and the number of records increase which would have effects on the response time as well, hence a more powerful server would be required to handle such requests. We will further discuss how do handle such situations and improve performance in chapter 7.

When the data was further analysed it was inferred that more then 40% of the contextual records for S4 were for WiFi information, although this can be useful but with several repetitions, it can be improved and made efficient. An approximate of 11% contextual records and 18% of

application usage data lacked location information which can be attributed to several factors such as unable to get a fix on GPS, provider being turned off, all the more reason to further investigated into the location obtaining process. All this presents us with room for improvement in the framework and the application as well.

Chapter 7

Challenges and concerns

As with any research, there are several challenging aspects and concerns surrounding this project that are to be considered both in terms of the present scope of the application/framework and in terms of the future prospects where the research can be used. In this chapter we briefly look into some of the key challenges faced by this project.

7.1 Platform restrictions and application deployment

The framework and the application in this report were both developed to be used on the Android operating system, this restricts the user who can participate in the study to only the one's with a smartphone which runs Android OS; users with other devices cannot contribute to the study in any way. This not only can cause an overall decrease in the number of participants, it can also effect the demographics and the quality of the data being collected as there can indeed be a variation in smartphone usage based on the operating system. Even though research (Gartner 2013) shows that Android holds a majority among the masses in terms of mobile marketshare, there are still a considerable number of users using other operating systems moreover the demographics of operating system usage can very widely across geographies. This presents a very serious concern and also a possible direction to look into in the future research as it is being faced by other researchers involved in similar crowd-sensing experiments (Balan, Nguyen, and Jiang 2011).

It should also be taken into consideration that Android is a very flexible platform which lets us build applications which can access such a wide array of information, some of which is currently not possible on other operating systems such as iOS. iOS, for example is governed by a sandbox policy (Apple Inc. 2013) which does not allow to application to access information beyond its assigned limit (such as extracting information about the running applications).

Being a crowd-sourcing application, the application is to be installed from an application store for android which presents yet another problem. This puts the decision of participation in the study solely on the fact that the user recognises the application on the store, installs and retains it on his device. The participation rate would vary depending on the application's popularity and would take some time to provide a considerable dataset which is explained further in (Xiao, Simoens, Pillai, Ha, and Satyanarayanan 2013), it also recommends on how to remove

installation of the application from the critical path but is presently not possible on Android for the type of information we require.

7.2 Resource utilisation

The purpose of the application is to monitor the how the user interacts with the smartphone and to observe the various contexts and sensors, which means that it has to quasi-continuously execute series of scans and poll sensors/API's in order to extract the required information. The extracted information has to be then stored on the disk and also has to be sent to the web server frequently for analysis using the internet connection of the device. All of these processes require a considerable amount of processing power, battery power and internet bandwidth (which in-turn means that the user may incur some monitory expense), hence making this a very critical aspect of the application which would have to be optimised.

For the application to be able to gather realistic information from the user it requires to be able to do it in an ubiquitous way, which would not be possible if the application is consuming too much of resources hence causing reduction in the average battery cycle of the smartphone or reducing the data bandwidth available to other apps which would make the connectivity an issue for other applications and causing the user experience of the applications to deteriorate.

Making the application perform all the required tasks while being efficient enough to have a very minimal footprint on the device was one of the key challenges of the project. Various steps were taken during development to make sure that the resource utilisation is kept to a minimum (see chapter 6 for detailed evaluation). For example, the application has an optimised synchronisation process, it prefers synchronisation of records on a WiFi connection rather then a cellular data connection, however if a threshold is reached in the collected information it tries to synchronise on cellular data network to reduce changes of data loss. The polling tasks of the framework are programmed to be functioned at fixed intervals which are specific to tasks and has a direct correlation with the type of information it provides, power consumption and required level of freshness of the information.The overall polling of context is done by the framework at certain fixed intervals each reflecting on the power consumption and the requirement for freshness of the data.

The resource utilisation can further be improved by developing the framework to perform adaptive sensing instead of scanning at fixed intervals. We can also do some pre-processing on the data (e.g., remove non required elements) hence reducing the overall bandwidth usage of the application. Deriving a relation between sensing rate and the power level of the device can help improve the battery life of the device. Sources of information which consume lesser power should be given preference for example using network instead of GPS for location detection is proven to save resources with an acceptable location accuracy (Zhuang, Kim, and Singh 2010).

7.3 Data privacy

One of the most common and possibly the most serious concern while developing a system that deals with user's personal information is privacy. Some argue that it is the system developer's

responsibility to take into consideration the privacy of the user and his information while designing the system (Adams and Sasse 2001). However it is essential to understand privacy concerns in these systems does not just end at ensuring restricted and authorised access of information through secure channels, there are several other technical and ethical aspects to it.

The application collects considerable amount of information about the end-user and stores it remotely, which allows a possibility for any external person to profile the user based on the information provided by third party which the user might not permit if asked for. It is not compulsory that the records have to be linked to the user for the breach to occur; inferences can be derived by using contextual information that is available along with external information sources to identify events and user activities (Minch 2004). Hence, keeping the information secure and impossible to use for identification from preying hands is very essential.

The first step which several usability studies mention in relation to such system's privacy aspect is to be transparent about the process that the application follows. Taking the user's consent and providing proper explanation about the reason for the collection of each and every contextual information from the user (Kelly 2006; Picard and Klein 2002); providing an insight on the data that is being collected and the subsequent use it would be put to allows the end-user to take a stand on whether or not he wishes to continue supplying data.

The possible use of the data gathered about user's activities, context and patterns can be to recommend information to the user in the future (see section 5.2). This may result in causing a subliminal behaviour changes which can very well be the purpose of the application as further demonstrated in various app mentioned in (Lathia, Pejovic, Rachuri, Mascolo, Musolesi, and Rentfrow 2013) however it can also be a repercussion of the app and may also cause a negative implication (for example, the application may recommend the user entertainment/social networking application while he is working based on his overall usage trends hence interrupting him and causing distraction). The control of how recommendations are provided or even how the application would be effecting the user's behaviour should hence be in the hands of the user and not the developer. In addition to this the application should also provide the user the ability to discontinue monitoring whenever he wants, which is taken care of in the form of a logout option in the application which terminates all monitoring tasks.

Chapter 8

Related Work

Over the last few years there have been several efforts in the form of researches that have worked on creating an extensible context sensing framework on various mobile platforms. Similarly there have been experiments conducted to gather information about user's smartphone usage and analyse mobile demographics in a general sense. In this chapter we shall briefly look into some of the more interesting and promising research work so far.

One of the most extensive work that has been done in the field of sensing on the mobile platform is the EmotionSense project[1]. The application is a part of a research whose goal was building a framework for mobile to assist in studies about user's emotions and behaviour. Previously developed for Nokia Symbian 60 platform (Rachuri, Musolesi, Mascolo, Rentfrow, Longworth, and Aucinas 2010), EmotionSense now has been re-modelled to work for the Android platform, several components of which have been open-sourced. There are several interesting components of the system which makes it very promising to use in any large scale mobile sensing project. The EmotionSense system constitutes of several sensor monitors which are of both polling and broadcast receiver model in nature. The inference engine present in the system provides the system the ability to adapt itself based on circumstances; for example monitoring of location only needs to happen when the user is moving (Rachuri, Musolesi, Mascolo, Rentfrow, Longworth, and Aucinas 2010). The sensor information is then stored in a knowledge base. In addition to this the framework (Symbian 60 version) also provides functionality for speaker and emotion recognition. However the android version presently provides only adaptive sensing and lacks methods for extraction of contextual information that are available on the more modern version of the operating system. Several researches which have made use of the framework have tried to extend it further and used it to build systems that can perform features like speaker recognition (Lathia, Pejovic, Rachuri, Mascolo, Musolesi, and Rentfrow 2013).

Another example for a sensing framework is the Mobile Sensing Framework (MSF) described in (Novak, Carlson, and Jarzabek 2013). The framework is designed for Android and makes use of Ambient Dynamics (Carlson and Schrader 2012) which enables it to dynamically adapt and control sensing after installation. The framework also enables to mine and extract historical context which can be used in various ways. *funf open sensing framework*[2] provides feature similar to MSF, however instead of letting user choose the way to store information the framework

[1] EmotionSense http://emotionsense.org/

[2] http://www.funf.org

provides an inbuilt integration with Dropbox[3] for automatically saving the information.

In terms of tracking and understanding user application usage there have been several studies and applications. One of the most promising one is FALCON (Yan, Chu, Ganesan, Kansal, and Liu 2012), a predictive app launching system which integrates with the Windows mobile operating system. Like *WebSense*, the application accesses mobile usage patterns of the user and combines it with location and other contextual parameters to predictively launch applications. The framework also takes into consideration the power, memory, launch time, energy and bandwidth factors prior launching the application and improves itself over the time (Yan, Chu, Ganesan, Kansal, and Liu 2012).

Another experiment (Do and Gatica-Perez 2010). Focused on building a framework to collect and mine large-scale patterns in mobile usage data presents various interesting patterns and statistics about mobile usage of a normal user for example it shows that user's tend to use majorly system application, primarily calling and SMS (which might be different in the past few years with new communication technologies). It also shows behaviour patterns among users and how several applications and why people interacts with it, are interrelated.

[3]http://www.dropbox.com/

Chapter 9

Future Work and conclusion

The application and the framework lays a foundation for a much more complex and useful anticipatory system through which not only can we understand more about the users but also help design future technologies to improve their experiences on mobile devices. Throughout this report we have discussed the various issues, shortcomings and have pointed out the scope for improvements in the present system. The framework presents us with a very simple yet effective approach to gather contextual data from smartphones of the user with the minimal development efforts for integration; providing the developers with much more time to concentrate on the real problem at hand rather then being concerned about data gathering and its efficiency. The contextual information that is being provided by the framework can be further improved both in terms of the quality and the details it can provide. Various other contextual sources can also be considered along with various other improvements that we have discussed in the section 3.4 in the future version of the framework along with some serious thoughts on the observations that we have made in chapter 6.

The android application which is at its initial stages of development, presently is extracting contextual information with the help of the framework and working alongside various other system components; however the data is at present being used for basic predictions tasks and application/web recommendations. As we have seen in section 3.4 the application would also require improvements both in terms of sensing techniques and in terms of the quality of information it provides us, information which could be harnessed to build systems and conduct further research;all of which shows us the potential of this application and the framework.

Mobile presence is not only a lifestyle anymore it has become a necessity, helping us in performing our day to day task and constantly helping us simplify our lives. It has already achieved a near-ubiquitous presence by blending itself into our lives and create an impact on it which we take for granted everyday. This has provided us an avenue for research in the forms of device capable of sensing information which was previously not possible without interrupting the user. Understanding how the user behaves and its relation to the context that surrounds him provides us an insight in creating a model for user's smartphone usage. This information empowers us in creating systems which can assist the user, help him make better choices or make him recommendations and even more bold and challenging ideas to create a better user experience in the future.

Bibliography

Adams, A. and M. A. Sasse (2001). Privacy in multimedia communications: Protecting users, not just data. In *People and Computers XVInteraction without Frontiers*, pp. 49–64. Springer.

Apple Inc. (2013). The iOS environment. `https://developer.apple.com/library/ios/documentation/iphone/conceptual/iphoneosprogrammingguide/TheiOSEnvironment/TheiOSEnvironment.html`. [Online; accessed 24-April-2014].

Balabanović, M. (1997). An adaptive web page recommendation service. In *Proceedings of the First International Conference on Autonomous Agents*, AGENTS '97, New York, NY, USA, pp. 378–385. ACM.

Balan, R. K., K. X. Nguyen, and L. Jiang (2011). Real-time trip information service for a large taxi fleet. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*, pp. 99–112. ACM.

Ben Abdesslem, F., A. Phillips, and T. Henderson (2009). Less is more: energy-efficient mobile sensing with senseless. In *Proceedings of the 1st ACM workshop on Networking, systems, and applications for mobile handhelds*, pp. 61–62. ACM.

Brown, M. G. (1996). Supporting user mobility. In *Mobile Communications*, pp. 69–77. Springer.

Campbell, A. T., S. B. Eisenman, N. D. Lane, E. Miluzzo, R. A. Peterson, H. Lu, X. Zheng, M. Musolesi, K. Fodor, and G.-S. Ahn (2008). The rise of people-centric sensing. *Internet Computing, IEEE 12*(4), 12–21.

Carlson, D. and A. Schrader (2012). Dynamix: An open plug-and-play context framework for android. In *Internet of Things (IOT), 2012 3rd International Conference on the*, pp. 151–158. IEEE.

Cheluvaraju, B., A. S. R. Kousik, and S. Rao (2011). Anticipatory retrieval and caching of data for mobile devices in variable-bandwidth environments. In *Systems Conference (SysCon), 2011 IEEE International*, pp. 531–537. IEEE.

Dede, E., M. Govindaraju, D. Gunter, R. S. Canon, and L. Ramakrishnan (2013). Performance evaluation of a mongodb and hadoop platform for scientific data analysis. In *Proceedings of the 4th ACM workshop on Scientific cloud computing*, pp. 13–20. ACM.

Dey, A. K. (2001). Understanding and using context. *Personal and ubiquitous computing 5*(1), 4–7.

Do, T.-M.-T. and D. Gatica-Perez (2010). By their apps you shall understand them: mining large-scale patterns of mobile phone usage. In *Proceedings of the 9th International Conference on Mobile and Ubiquitous Multimedia*, pp. 27. ACM.

Ducatel, K., M. Bogdanowicz, F. Scapolo, J. Leijten, and J.-C. Burgelman (2001). *Scenarios for ambient intelligence in 2010*. Office for official publications of the European Communities.

Dutta, P., P. M. Aoki, N. Kumar, A. Mainwaring, C. Myers, W. Willett, and A. Woodruff (2009). Common sense: participatory urban sensing using a network of handheld air quality monitors. In *Proceedings of the 7th ACM conference on embedded networked sensor systems*, pp. 349–350. ACM.

Ganti, R. K., F. Ye, and H. Lei (2011). Mobile crowdsensing: current state and future challenges. *Communications Magazine, IEEE 49*(11), 32–39.

Gartner (2013). Sales of mobile phones in all regions except asia/pacific declined in the first quarter of 2013. `http://www.gartner.com/newsroom/id/2482816`. [Online; accessed 24-April-2014].

Google (2014a). Android activity lifecycle. `http://developer.android.com/reference/android/app/Activity.html#ActivityLifecycle`. [Online; accessed 24-April-2014].

Google (2014b). Android dalvik debug monitor server (ddms). `http://developer.android.com/tools/debugging/ddms.html`. [Online; accessed 24-April-2014].

Ho, J. and S. S. Intille (2005). Using context-aware computing to reduce the perceived burden of interruptions from mobile devices. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 909–918. ACM.

Hull, B., V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. Miu, E. Shih, H. Balakrishnan, and S. Madden (2006). Cartel: a distributed mobile sensor computing system. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, pp. 125–138. ACM.

Kelly, K. (2006). Symmetrical and Asymmetrical Technologies. `http://kk.org/thetechnium/2006/02/symmetrical-and/`. [Online; accessed 24-April-2014].

Lane, N. D., E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. T. Campbell (2010). A survey of mobile phone sensing. *Communications Magazine, IEEE 48*(9), 140–150.

Lathia, N., V. Pejovic, K. K. Rachuri, C. Mascolo, M. Musolesi, and P. J. Rentfrow (2013). Smartphones for large-scale behaviour change interventions. *IEEE Pervasive Computing (May 2013) 1*, 3–9.

Lu, H., D. Frauendorfer, M. Rabbi, M. S. Mast, G. T. Chittaranjan, A. T. Campbell, D. Gatica-Perez, and T. Choudhury (2012). Stresssense: Detecting stress in unconstrained acoustic environments using smartphones. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pp. 351–360. ACM.

Minch, R. P. (2004). Privacy issues in location-aware mobile devices. In *System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on*, pp. 10–pp. IEEE.

Mohan, P., V. N. Padmanabhan, and R. Ramjee (2008). Nericell: rich monitoring of road and traffic conditions using mobile smartphones. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pp. 323–336. ACM.

MongoLab (2013). Deep dive into connection pooling. `http://blog.mongolab.com/2013/11/deep-dive-into-connection-pooling/`. [Online; accessed 24-April-2014].

Novak, G., D. Carlson, and S. Jarzabek (2013). An extensible mobile sensing platform for mhealth and telemedicine applications. In *Proceeding of Conference on Mobile and Information Technologies in Medicine (MobileMed 2013), At Prague, Czech Republic*.

Oulasvirta, A., S. Tamminen, V. Roto, and J. Kuorelahti (2005). Interaction in 4-second bursts: the fragmented nature of attentional resources in mobile hci. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 919–928. ACM.

Pejovic, V. and M. Musolesi (2013). Anticipatory mobile computing: A survey of the state of the art and research challenges. *arXiv preprint arXiv:1306.2356*.

Picard, R. W. and J. Klein (2002). Computers that recognise and respond to user emotion: theoretical and practical implications. *Interacting with computers 14*(2), 141–169.

Prekop, P. and M. Burnett (2003). Activities, context and ubiquitous computing. *Computer Communications 26*(11), 1168–1176.

Rachuri, K. K., M. Musolesi, C. Mascolo, P. J. Rentfrow, C. Longworth, and A. Aucinas (2010). Emotionsense: a mobile phones based adaptive platform for experimental social psychology research. In *Proceedings of the 12th ACM international conference on Ubiquitous computing*, pp. 281–290. ACM.

Rosen, R. (2012). *Anticipatory systems*. Springer.

Ross, P. E. (2011). Top 11 technologies of the decade. *IEEE Spectrum 48*(1), 27–63.

Sadri, F. (2011). Ambient intelligence: A survey. *ACM Computing Surveys (CSUR) 43*(4), 36.

Schilit, B., N. Adams, and R. Want (1994). Context-aware computing applications. In *Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on*, pp. 85–90. IEEE.

Schilit, B. N. and M. M. Theimer (1994). Disseminating active map information to mobile hosts. *Network, IEEE 8*(5), 22–32.

Tennenhouse, D. (2000). Proactive computing. *Communications of the ACM 43*(5), 43–50.

Tilkov, S. and S. Vinoski (2010). Node. js: Using javascript to build high-performance network programs. *IEEE Internet Computing 14*(6).

Weiser, M. (1991). The computer for the 21st century. *Scientific american 265*(3), 94–104.

Xiao, Y., P. Simoens, P. Pillai, K. Ha, and M. Satyanarayanan (2013). Lowering the barriers to large-scale mobile crowdsensing. In *Proceedings of the 14th Workshop on Mobile Computing Systems and Applications*, pp. 9. ACM.

Yan, T., D. Chu, D. Ganesan, A. Kansal, and J. Liu (2012). Fast app launching for mobile devices using predictive user context. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pp. 113–126. ACM.

Zhuang, Z., K.-H. Kim, and J. P. Singh (2010). Improving energy efficiency of location sensing on smartphones. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pp. 315–330. ACM.

Appendix A

Mini-Project Declaration

The University of Birmingham

School of Computer Science

Second Semester Mini-Project: Declaration

1. Project Details

Name: Karthikeya Udupa Kuppar Manjunath

Student number: 1393456

Mini-project title: *Analysing User Web Interaction On Mobile Devices.*
Mini-project supervisor: Mirco Musolesi <m.musolesi@cs.bham.ac.uk>

2. Project Description

The following questions should be answered in conjunction with a reading of your programme handbook.

Aim of mini-project	*To capture and analyse the way in which the user interacts with his device in relation to his context.*

Objectives to be achieved	• *Development of a methodology to capture user's interaction with his mobile device.*
	  • *Identifying the URL's being accessed by the user.*
	  • *Tracking the application's usage by the user.*
	• *Identifying the various context's associated with the user during the interaction.*
	  • *Finding the location of the user when the content was accessed along with time.*
	  • *Uniquely identifying the user amongst all other users.*
	  • *Tagging of important location with respect to the user.*
	• *Storing and categorisation of the information that is collected from the user.*
	  • *Identifying the nature of the content on the website.*
	  • *Identifying the nature of the application that is being used.*
	• *Analysing the possible use of such information in development of anticipatory systems.*
	• *Literature review of anticipatory systems.*

Project management skills **Briefly explain how you will devise a management plan to allow your supervisor to evaluate your progress**	*The initial 4 weeks period would be dedicated to understanding the various possible approaches to collect the various possible information from the user.* *A two week period would be to analyse the information to build an effective application for gathering of information (in android platform).* *The reminder of the time would be given to analysing the information that has been collected and understanding the possible usage and also in perfection of the application.* *To Summarise the plan.* • ***Week 1-4:*** *Understanding the approaches to extract the various possible information from the user.* • ***Week 5-6:*** *Development of an refined application to collect and display the relevant information.* • ***Week 6-9:*** *Formulating and finalising a report concluding the findings and possible applications of information that was collected.*

Systematic literature skills **Briefly explain how you will find previous relevant work**	*The primary source of the previously done relevant work would be through the library and the online research/conference paper archives.* *Further information would also be gained by interacting with the various people working in the related fields.*

Communication skills **What communication skills will you practise during this mini-project?**	*A bi-weekly meeting with the mini-project supervisor to discuss and update the progress of the project with intermediate updates through email.* *Email would be used as a primary means to contact the supervisor in case of any requirement of assistance.*

3. Project Ethics Self-Assessment Form

Appendix B

Statement of information search strategy

B.1 Parameters for literature search

Forms of literature: The following literature sources were reviewed:

- Conference papers
- Journal articles
- Theses
- Books
- API Documentation

Geographical/language coverage: Reference papers used for this work is were from North America and Western Europe. Preferred language for all the reference paper was English.

B.2 Appropriate search tools

- *Google Scholar*: Was used to retrieve conference papers, journal articles and some theses. This search engine was used for a wide coverage of research topics and experiments conducted in the area of anticipatory computing, context sensing, user behaviour monitoring among others.
- *Android SDK Documentation*: The android SDK was used to understand the operating system's functioning and its various components.

Appendix C

Configurations

```
public static long BG_SERVICE_ALARM_RELOAD_TIME = 25*1000L; //service revival time.
public static long TASK_POLLER_TIMER = 4*1000L;
public static int MIN_RECORD_FOR_SYNC = 20; //for WiFi
public static int RECORD_THRESHOLD_FOR_FORCED_SYNCED = 100; //on cellular.
public static int RECORD_BATCH_COUNT = 50;
```

Snippet 3: Constants for the *WebSense* application.

```
mContextManager.monitorContext(
ContextManagerServices.CTX_FRAMEWORK_LOCATION, 60*60*1000L);
mContextManager.monitorContext(
ContextManagerServices.CTX_FRAMEWORK_BATTERY, 10*60*1000L);
mContextManager.monitorContext(
ContextManagerServices.CTX_FRAMEWORK_WIFI, 30*60*1000L);
mContextManager.monitorContext(
ContextManagerServices.CTX_FRAMEWORK_SIGNALS, 15*60*1000L);
mContextManager.monitorContext(
ContextManagerServices.CTX_FRAMEWORK_EVENTS, 2*60*1000L);
mContextManager.monitorContext(
ContextManagerServices.CTX_FRAMEWORK_BLUETOOTH, 10*60*1000L);
```

Snippet 4: Initialisation for sensing from *WebSense* with minimum broadcast timing specified.

```
public static String DEFAULT_LOCATION_PROVIDER = NETWORK_PROVIDER;
public static long MAXIMUM_ACCEPTABLE_TIME = 40*1000L;
public static long MAXIMUM_ACCEPTABLE_DISTANCE = 0;
public static long SHORT_POLLING_INTERVAL =  15*60*1000L;
public static final long MINUTE_POLLING_INTERVAL = 3*60*1000L;
public static final long LONG_POLLING_INTERVAL = 30*60*1000L;
public static final long GRANULAR_POLLING_INTERVAL = 10*1000L;
```

Snippet 5: Constants for the Android context sensing framework.